Технології програмування

Лабораторна робота №2

**Робота з файлами. Юніт тести.**

**Мета роботи:** Використовуючи теоретичне підґрунтя про роботу з файлами та тестування коду у мові Python розширити програму телефонного довідника студентів додавши функціонал, що буде вказано в завданні до лабораторної роботи.

**Завдання до лабораторної роботи**

Розширити функціональність телефонного довідника студентів групи, що був розроблений у Лабораторній роботі №1 наступними можливостями:

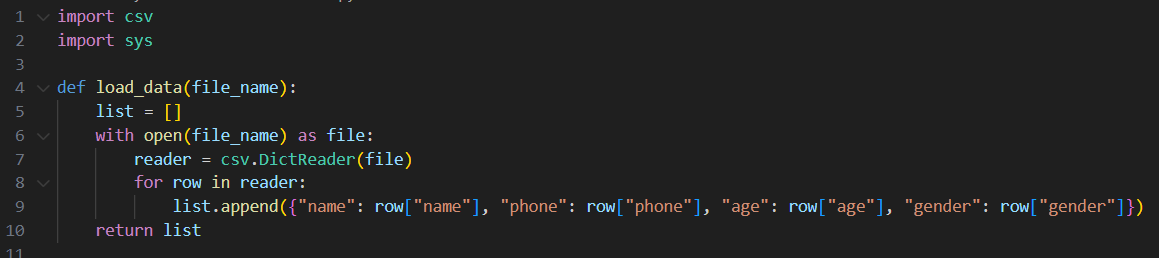
1) завантаження початкових данні для довідника з csv файлу. Приклад csv файлу (lab2.csv), та оригінальній файл формату xlsx розміщені поряд з вказівками. Ім’я файлу для завантаження даних повинно передаватись в якості параметрів командного рядку.

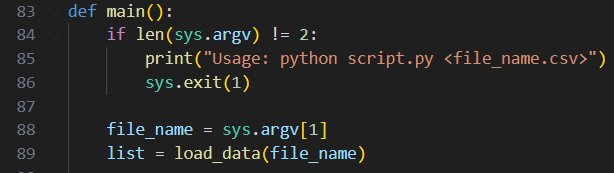
2) зберігання довідника студентів у форматі CSV перед виходом із програми.

3) покриття основного функціоналу програми Юніт тестами.

**Хід роботи**

1. Завантаження початкових даних для довідника з csv файлу.
   * Робимо функцію завантаження даних.
   * Створюємо порожній список, який буде використовуватися для зберігання словників даних.
   * Відкриваємо файл з ім'ям, що міститься в змінній file\_name, для читання. Використовуємо конструкцію with, щоб автоматично закрити файл після виконання блоку коду.
   * Створюємо об'єкт DictReader, який дозволяє читати рядки CSV-файлу як словники, використовуючи перший рядок як назви стовбців.
   * Проходимось по кожному рядку CSV-файлу.
   * Для кожного рядка створюємо словник, де ключі відповідають назвам стовбців ("name", "phone", "age", "gender"), а значення відповідають відповідним значенням з поточного рядка CSV.
   * Повертаємо список.
   * В основну функцію додаємо умову, якщо кількість аргументів, що передаються не дорівнює 2, то виводимо відповідний напис. Повертаємо помилку.
   * Вказуємо змінну file\_name, що дорівнює другому аргументу, який вводить користувач.
   * Вказуємо змінну list, що дорівнює функції, що створює список з csv файлу.

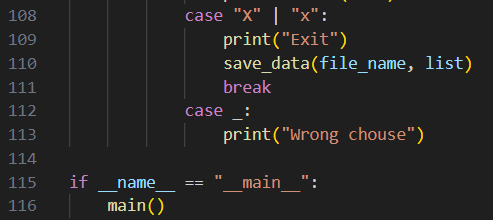




1. Зберігання довідника студентів у форматі CSV перед виходом із програми.

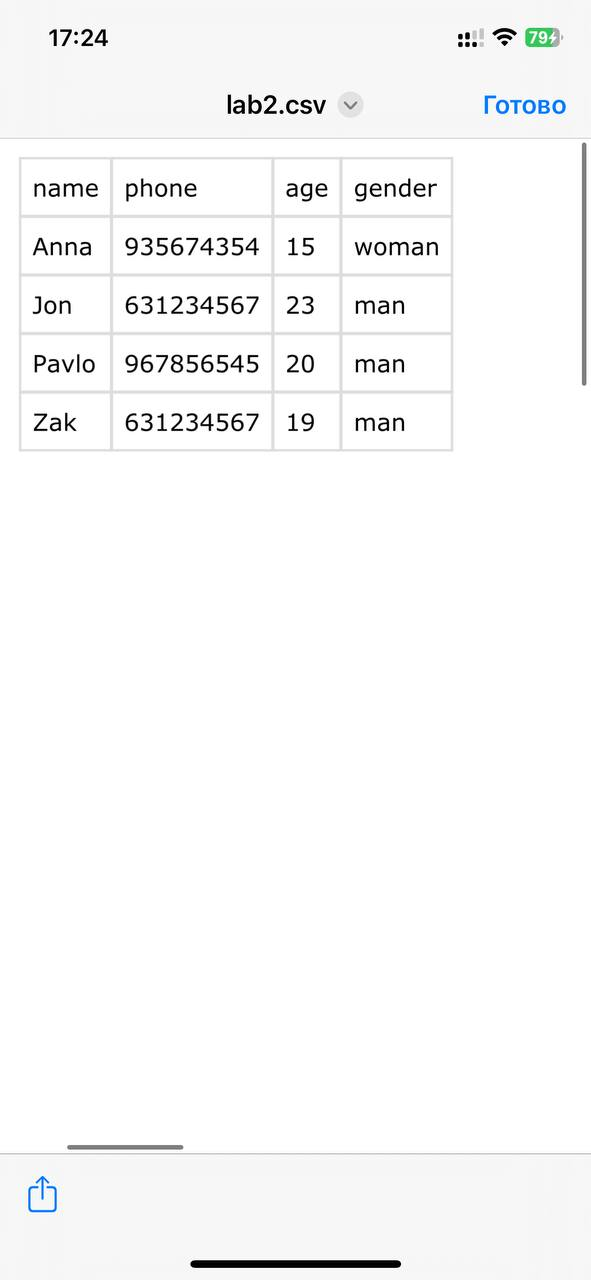
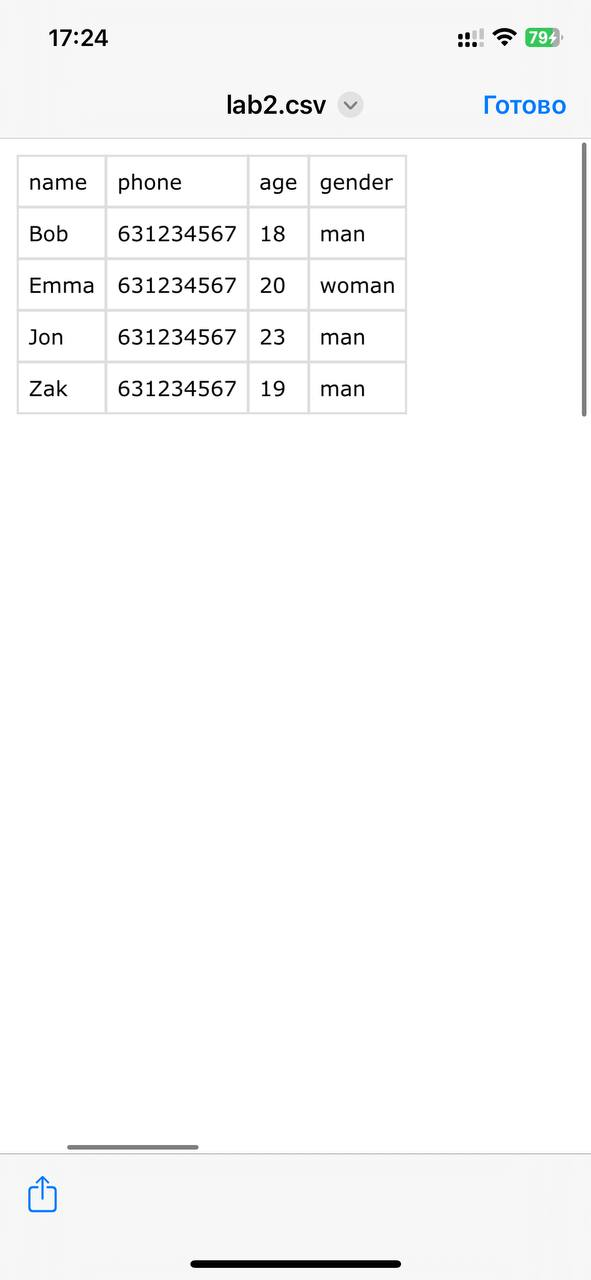
* Робимо функцію зберігання даних.
* Створюємо список fieldnames, який містить імена полів у порядку, в якому вони будуть виведені у файлі CSV.
* Відкриваємо файл з ім'ям, що міститься в змінній file\_name, для запису у текстовому режимі. Параметр newline дозволяє уникнути автоматичного додавання додаткового символу нового рядка при записі.
* Створюємо об'єкт DictWriter, який дозволяє нам записувати словникові дані у файл CSV. Вказуємо імена полів за допомогою параметра fieldnames.
* Записуємо рядок заголовка у файл CSV, що містить імена полів.
* Записуємо всі рядки зі списку list у файл CSV.
* Додаємо виконання цієї функції при вводі користувача X або x.





1. Вивід готового коду в терміналі та зміна файлу після виходу з програми.

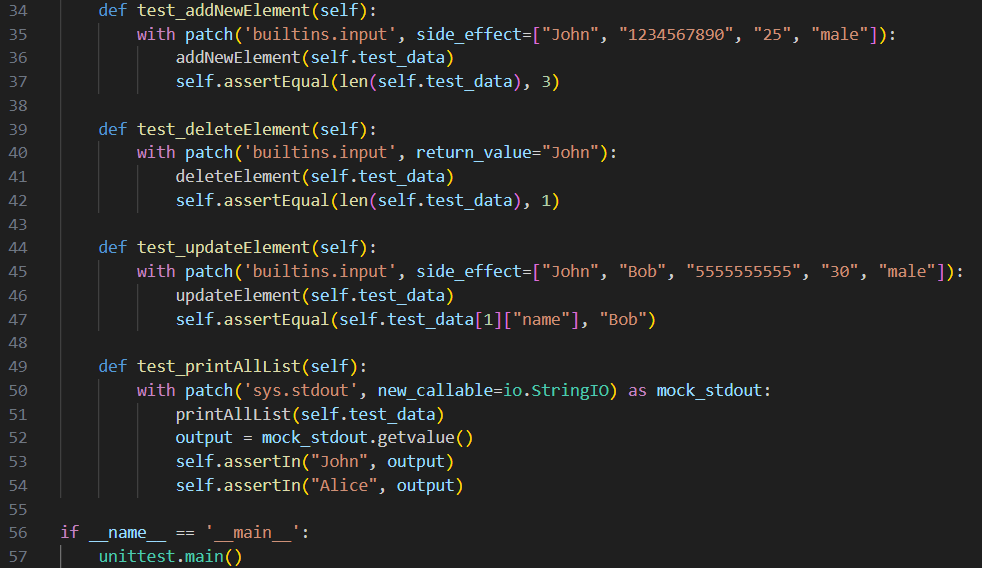


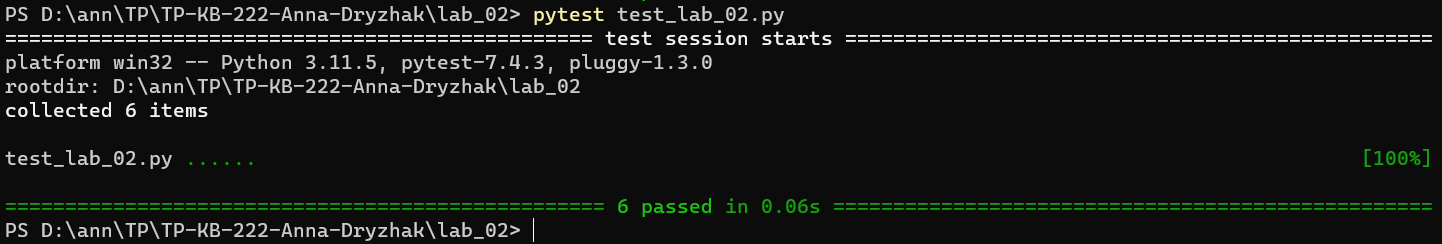


1. Покриття основного функціоналу програми Юніт тестами.

* Імпортуємо модуль unittest, який надає інфраструктуру для написання тестів.
* Імпортуємо модуль io, який надає інструменти для роботи з потоками вводу/виводу.
* Імпортуємо функціонал patch з модуля unittest.mock. Він використовується для заміни частин коду при тестуванні, щоб імітувати певні дії.
* Імпортуємо всі наші функції з файлу lab\_02.py.
* Створюємо клас тестів, який наслідується від unittest.TestCase.
* Метод setUp викликається перед кожним тестом у класі. Він містить дві змінні: test\_file (ім'я тестового файлу) і test\_data (тестові дані у вигляді списку словників).
* Створюємо перевірку функції load\_data.
* Використовуємо open для створення тестового файлу (test\_file) і запису в нього деяких даних у форматі CSV.
* Потім викликаємо функцію load\_data для завантаження даних з цього тестового файлу.
* Викликаємо self.assertEqual(data, [...]), щоб порівняти отримані дані з очікуваними, а саме, що data буде списком словників, які відповідають даним, які ми записали в тестовий файл.
* Створюємо перевірку функції save\_data.
* Викликаємо функцію save\_data(self.test\_file, self.test\_data). Ця функція повинна зберігати дані (self.test\_data) у вказаний файл (self.test\_file).
* Потім завантажуємо дані з цього файлу за допомогою функції load\_data(self.test\_file) і зберігаємо їх у змінну loaded\_data.
* Викликаємо self.assertEqual(loaded\_data, self.test\_data), щоб порівняти завантажені дані з оригінальними.
* Створюємо перевірку функції addNewElement.
* Використовуємо patch('builtins.input', side\_effect=["John", "1234567890", "25", "male"]), щоб замінювати введення користувача під час виконання.
* Викликаємо функцію addNewElement(self.test\_data). Ця функція приймає дані від користувача і додає новий елемент у список даних.
* Після цього перевіряємо, чи довжина списку (len(self.test\_data)) збільшилася на одиницю. Це вказує на те, що новий елемент був доданий до списку.
* Аналогічно робимо перевірку для функцій deleteElement та updateElement.
* Створюємо перевірку функції printAllList..
* Використовуємо patch('sys.stdout', new\_callable=io.StringIO) для того, щоб замінити стандартний вивід (sys.stdout) на io.StringIO, який є об'єктом для роботи з рядками як з файлами. Весь вивід, який потрапив би у стандартний вивід, тепер буде перехоплено і збережено у StringIO. Це дозволяє отримати вміст, який виводить функція printAllList, замість того, щоб виводити його на екран.
* Викликаємо функцію printAllList(self.test\_data), яка виводить дані у консоль.
* Викликаємо mock\_stdout.getvalue(), щоб отримати рядок, який був виведений у консоль.
* Після цього перевіряємо, чи ім'я John та Alice присутні у виведеному рядку.





1. Результат перевірки. 

**Висновок.** Під час виконання даної лабораторної роботи я використовуючи теоретичне підґрунтя про роботу з файлами та тестування коду у мові Python розширила програму телефонного довідника студентів додавши функціонал, що було вказано в завданні до лабораторної роботи.